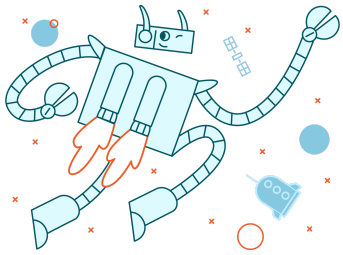
Functions



Functions are like containers for code. They carry instructions to perform a particular task.

Function Definition

To define a function in JavaScript, we use the `function` keyword. For example,

```
function functionName(parameter) {
    // function code
}
```

Return Values

We can end a function execution and specify a result from a function by using the `return` keyword. For example,

```
function add(parameter1, parameter2) {
    return parameter1 + parameter2
}
```

We can also pass this return values to other functions. For example, passing the return value of `add` function to the `alert` function.

```
function add(lhs, rhs) {
    return lhs + rhs
}

alert(String(add(1014, 323)))
```

Function Call

Once our function is defined, we can simply call the function just by its function name. For example,

```
// Function defintion
function add(para1, para2){
    let addition = para1 + para2
    return addition
}

// Function call
let result = add(4, 5)
console.log(result)
```

Console:

```
9
```

We can also call functions within other functions. For example,

```
// Function defintion
function add(para1, para2){
    let addition = para1 + para2
    return addition
}

function addTenToEverything(para1, para2){
    let addition = add(para1, para2)
    return addition + 10
}

let result = addTenToEverything(4, 5)
console.log(result)
```

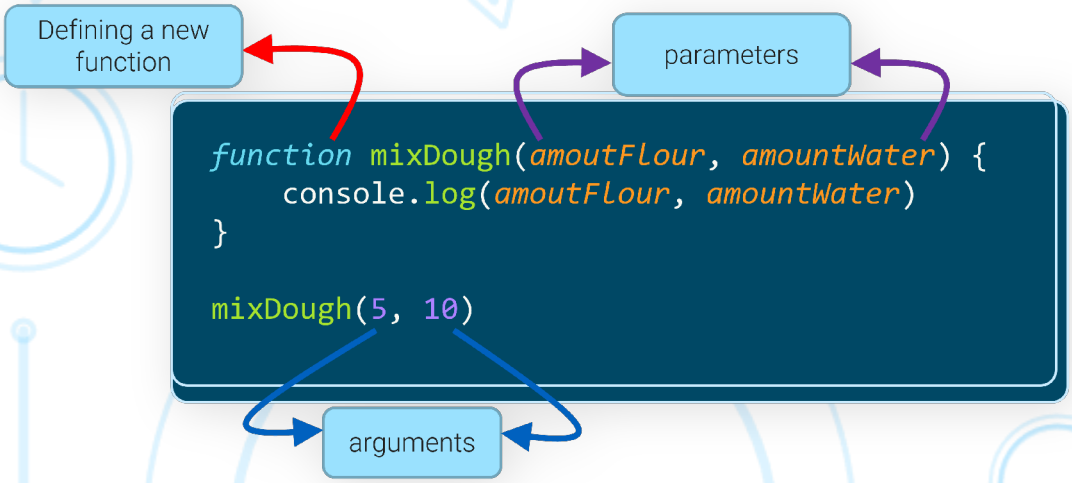
Console:

```
19
```

Parameters and Arguments

A **parameter** is the variable listed inside the parantheses in the function defintion.

An **argument** is the value that is sent to the function when it is called.



Scopes

Local Variables

When we define a variable **within** a function definition, the variable has its scope in the function block for which it is declared. Thus, it is a **local** variable. For example,

```
function doA() {
    // Local variable
    let a = 3
}

doA()
```

The variable, `a` is now a local variable with the value 3 stored.

Global Variables

On the other hand, when you define a variable **outside** the function definition, the variable has its scope in the entire program. Thus, it becomes

a **global** variable. For example,

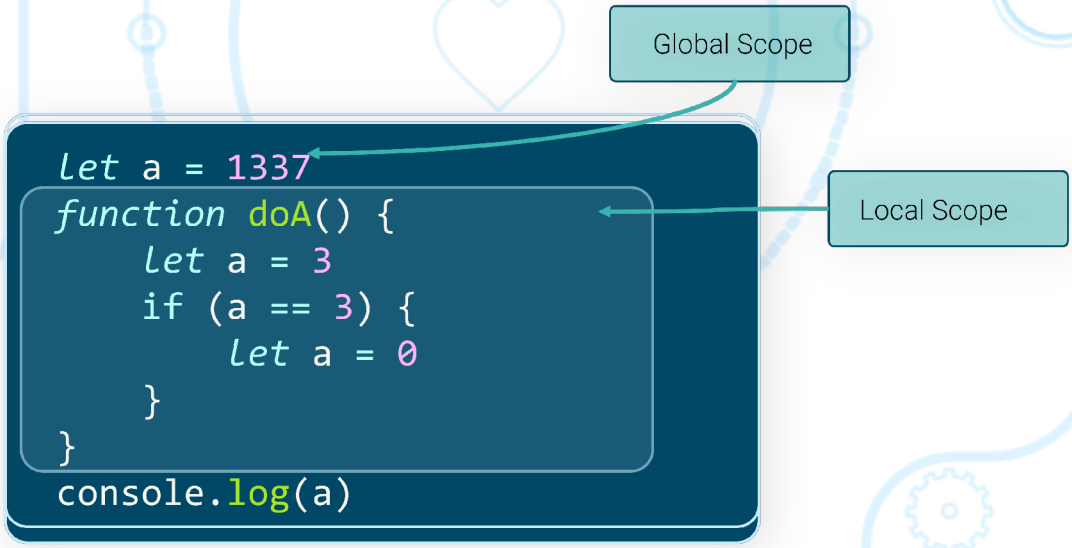
```
// Global variable
let a = 1337

function doA() {
    a = 3
    console.log(a)
}

doA()
```

The variable, `a` is now a global variable with the value 3 stored.

Local vs Global Scopes



- The variable `a` declared **inside** the `doA()` function can only be accessed within this function. As such any change in the local variable does not affect other functions of the program.
- On the other hand, the variable `a` declared **outside** the function can be accessed globally in the entire program and by all functions present in the program. Thus, any change in the global variable will affect the whole program, wherever it is being used.

Abstraction

Functions encapsulate the idea of **abstraction**. Using a single command, we are able to carry out a complicated task without knowing the implementation details.

For example, the `arithmeticOperations` function below carries out 4 steps: addition, multiplication, division and subtraction. Instead of performing these 4 steps, we can use a single step by calling the `arithmeticOperations` function.

```
function arithmeticOperations(number){
    num += 10
    num *= 20
    num /= 30
    num -= 40
    return num
}

let result = arithmeticOperations(8)
console.log(result)
```

Console:

```
-28
```

The use of functions allows us to produce code that is **readable**, **modular**, easier to **extend** and easier to **debug**.

Further Reading

Block Scope

A block scoped variable is a variable that will not be accessible from outside the block which it is defined in.

```
let a = 3

// block 1
if (a == 3) {
    let a = 1337
}

console.log(a)
```

Output:

```
3
```

In the example above, variable `a (block 1)` is declared inside the `if` block and cannot be accessed outside this block.

Name	Value
a	3
a (block 1)	1337

Declaring Variables

In unit 1, we learnt how to declare our variables using `var`. In this unit, we will be declaring our variables by using the `let` statement.

```
let a = 3
```

Let vs Var

`let` provides block scope in JavaScript, but variables declared with `var` use **function scope** meaning they are accessible from anywhere inside the function. Thus variables declared by `var` inside a `{}` (e.g. if) block can be accessed from outside the block.

```
if (true) {
  let x = 2;
}

// Fails, x can NOT be used here
console.log(x)
```

```
if (true) {
  var x = 2;
}

// Works, x CAN be used here
console.log(x)
```

For more information about the difference between `let` and `var` you can visit this link